Lab Assignment 10: Convolutional Neural Networks with Python

Shiva Agarwal

May 16, 2024

Lab Assignment

Coding Question

You are required to implement a Convolutional Neural Network (CNN) using Python with Keras library for classifying images from the CIFAR-10 dataset. Follow the steps below:

1. Import the CIFAR-10 dataset and print training data-related information.
2. Import the necessary libraries for building and training the CNN model.
3. Specify the initial convolutional block of the CNN.
4. Add dropout layer for regularization.
5. Add another convolutional block for CNN.
6. Define dense layers that consume the feature array and produce a classification.
7. Compile and train the CNN model.
8. Evaluate the performance of the trained model.
9. Check for overfitting using appropriate visualization.
10. Make predictions using the trained model.
11. Print the confusion matrix to evaluate the model's performance.

Solution

Question 1: Importing Libraries

```
import numpy as np
import matplotlib.pyplot as plt
from keras.callbacks import EarlyStopping
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Flatten
from keras.layers import LeakyReLU
from keras.layers.convolutional import Conv2D
from keras.optimizers import Adam
from keras.layers.pooling import MaxPooling2D
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix
```

Question 2: Loading the CIFAR-10 Dataset

```
# Load CIFAR-10 dataset
(X_train, Y_train), (X_test, Y_test) = cifar10.load_data()

# Print training data-related information
print("Shape of training data:", X_train.shape)
print("First 5 labels:", Y_train[:5])
```

Question 3: Building the CNN Model

```
# Build CNN model
classifier = Sequential()

classifier.add(Conv2D(32, kernel_size=(3, 3), padding='same', input_shape=(32, 32, 3)))
classifier.add(LeakyReLU(alpha=0.3))
classifier.add(Conv2D(64, padding='same', kernel_size=(3, 3)))
classifier.add(LeakyReLU(alpha=0.3))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Dropout(0.25))

classifier.add(Conv2D(128, kernel_size=(3, 3)))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Conv2D(128, kernel_size=(3, 3)))
classifier.add(LeakyReLU(alpha=0.3))
classifier.add(MaxPooling2D(pool_size=(2, 2)))
classifier.add(Dropout(0.25))

classifier.add(Flatten())
classifier.add(Dense(1024))
classifier.add(LeakyReLU(alpha=0.3))
classifier.add(Dropout(0.5))
classifier.add(Dense(10, activation='softmax'))
```

Question 4: Compiling and Training the CNN Model

```
# Compile CNN model
classifier.compile(loss='categorical_crossentropy',
                   optimizer=Adam(lr=0.0001, decay=1e-6),
                   metrics=['accuracy'])

# Train CNN model
history = classifier.fit(X_train / 255.0, to_categorical(Y_train),
                         batch_size=128, shuffle=True,
                         epochs=250,
                         validation_data=(X_test / 255.0, to_categorical(Y_test)),
                         callbacks=[EarlyStopping(min_delta=0.01, patience=4)])
```

Question 5: Evaluating the Trained Model

```
# Evaluate CNN model
scores = classifier.evaluate(X_test / 255.0, to_categorical(Y_test))
print('Loss:', scores[0])
print('Accuracy:', scores[1])

# Check for overfitting
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model␣Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper␣left')
plt.show()
```

Question 6: Making Predictions

```
# Make predictions
sample_id = 108
plt.figure(figsize=(4, 4))
plt.imshow(X_test[sample_id])
plt.axis('off')
plt.show()

print("Predicted class:", np.argmax(classifier.predict(
    X_test[108].reshape((1,) + X_test[108].shape) / 255.))
    )
```

Question 7: Printing the Confusion Matrix

```
# Print confusion matrix
Y_preds = classifier.predict(X_test / 255.0)
cm = confusion_matrix(Y_test, Y_preds.argmax(axis=1))
print("Confusion Matrix:")
print(cm)
```